|  |  |  |  |
| --- | --- | --- | --- |
| **STUDENT USE** | | **STAFF USE** | |
| Module Name | Object Oriented Programming | First Marker’s  (acts as signature) |  |
| Module Code | 5COSC018C-n | Second Marker’s  (acts as signature) |  |
| Lecturer Name | Avazkhan Khalikov | Agreed Mark |  |
| UoW Student IDs |  | **For Registrar’s office use only (hard copy submission)** | |
| WIUT Student IDs | 00010443 |
| Deadline date | 17/12/2021 |
| Assignment Type | 🗌Group☑Individual |

**General Introduction.**

Case 8.

University decided to recruit new hires as candidates for Human Resources.

To do this, they contacted our Application creation service with front-end and back-end functions. Therefore, as a software architecture, I take responsibility for creating an application where users can do the following:

1. Access to personnel or candidate based on professions.

2. Creation of own information by the candidate.

3. Personnel list of all announced candidates.

The university applies modern practices of personnel management. Working in these markets, any organization uses both professional practices and the science of decision-making in these markets.

In turn, professional practices are used in personnel management:

- Personnel administration, HR workflow and accounting systems (HR administration, HR data management);

- Recruitment (search and selection) (staffing, recruitment, talent acquisition);

- Learning and development (learning and development, training and development);

- Remuneration (compensation management, total rewards).

The main users of the application will be managing institutions and human resources management.

Case 5.

An optimized transport service, that is, an online taxi ordering company needs an application to manage taxi call orders.

The application is required to have features such as:

1. Map display

2. The customer order form from the taxi company worker is displayed

3. List of orders of the taxi driver with actions for taking orders.

This application is the main functional of the technical work of the company, where working with clients will be managed.

Case 9.

The street billboard management system needs an application to control the location of banners. For this, the application must have such functions as:

1. The function of clicking on the point where the obtained coordinates are transmitted to the backend service

2. Search by banner coordinates

3. Development of CRUD Banner operations

The main requirement of the application's functionality is to work with maps based on the location of the banners.

These three projects to create applications using specific design patterns should be carried out by our team of software architects.

**Main Task**

**1. Intro**

Case 8.

Access to personnel management can only be used by human resources management and university management. They can search and find candidates, remove or accept them, change them by profession. For developing this we can use the design pattern Builder.

Builder - A design pattern that encapsulates the creation of an object and allows it to be divided into different stages.

When to use the Builder pattern?

- When the process of creating a new object should not depend on what parts this object consists of and how these parts are related to each other

- When it is necessary to ensure that different variations of an object are obtained during its creation

Case 5.

The application for ordering a taxi is used by the company administration, drivers, clients. The driver can receive orders, and customers can create or then cancel these orders, and both use map through the application. Customers should not have access to other client's orders.

The Strategy Pattern is used for this.

The Strategy design pattern defines a family of algorithms, encapsulate each one, and make them interchangeable. This pattern lets the algorithm vary independently from clients that use it.

In other words, this pattern represents a design pattern that defines a set of algorithms, encapsulates each of them, and makes them interchangeable. Depending on the situation, we can easily replace one used algorithm with another. In this case, the replacement of the algorithm occurs regardless of the object that uses this algorithm.

Case 9.

The application is accessed by the Street Banner management. Managers must have access to the location of all Billboards and the values of their coordinates. For this it is used The Iterator pattern, which provides an abstract interface for sequentially accessing all the elements of a composite object without exposing its internal structure.

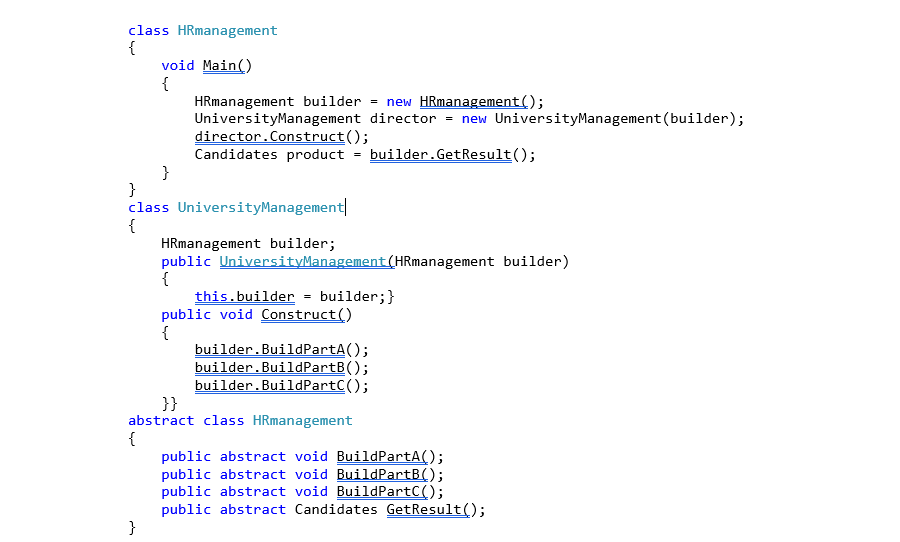
When to use iterators?

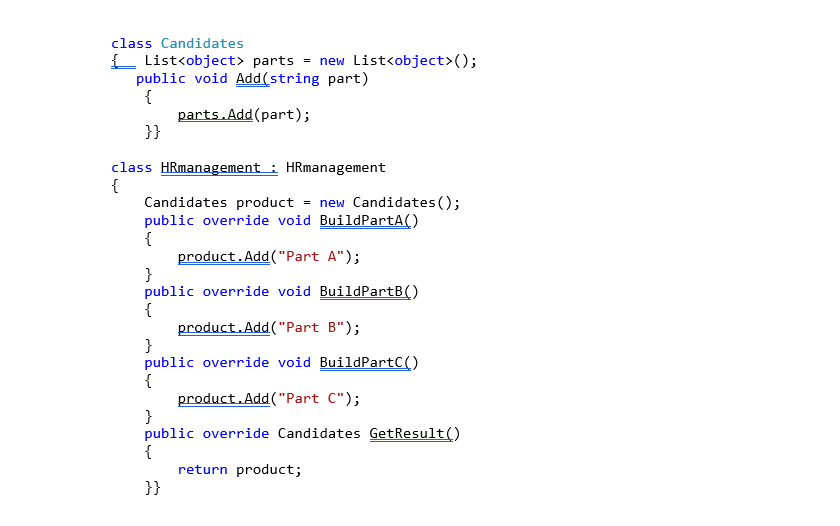
- When it is necessary to traverse an object without revealing its internal structure

- When there is a set of complex objects, and it is necessary to provide a single interface for enumerating them

- When it is necessary to provide several alternative options for iterating over the same object

Case 8

****

****

Case 5

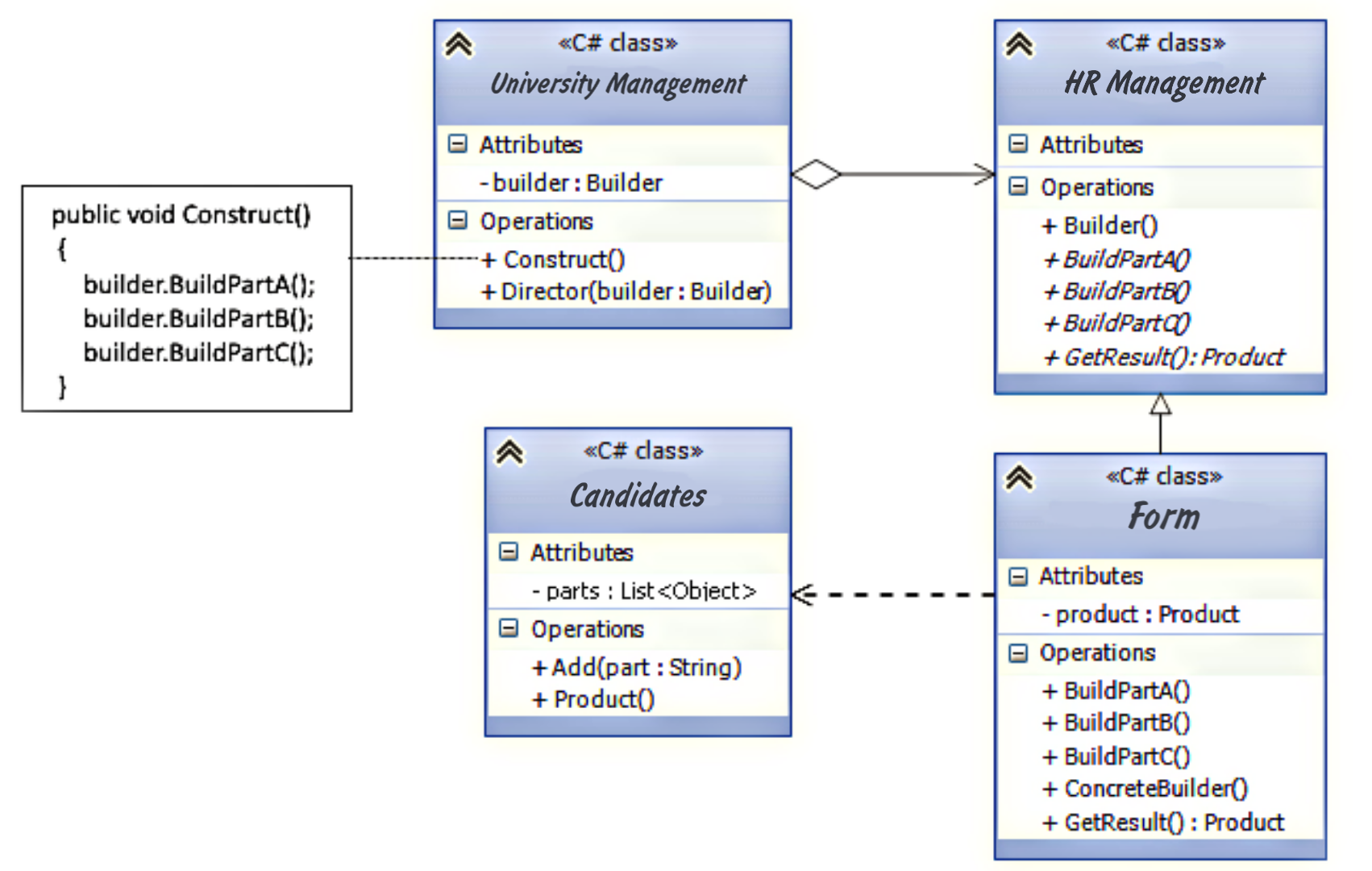


Case 9

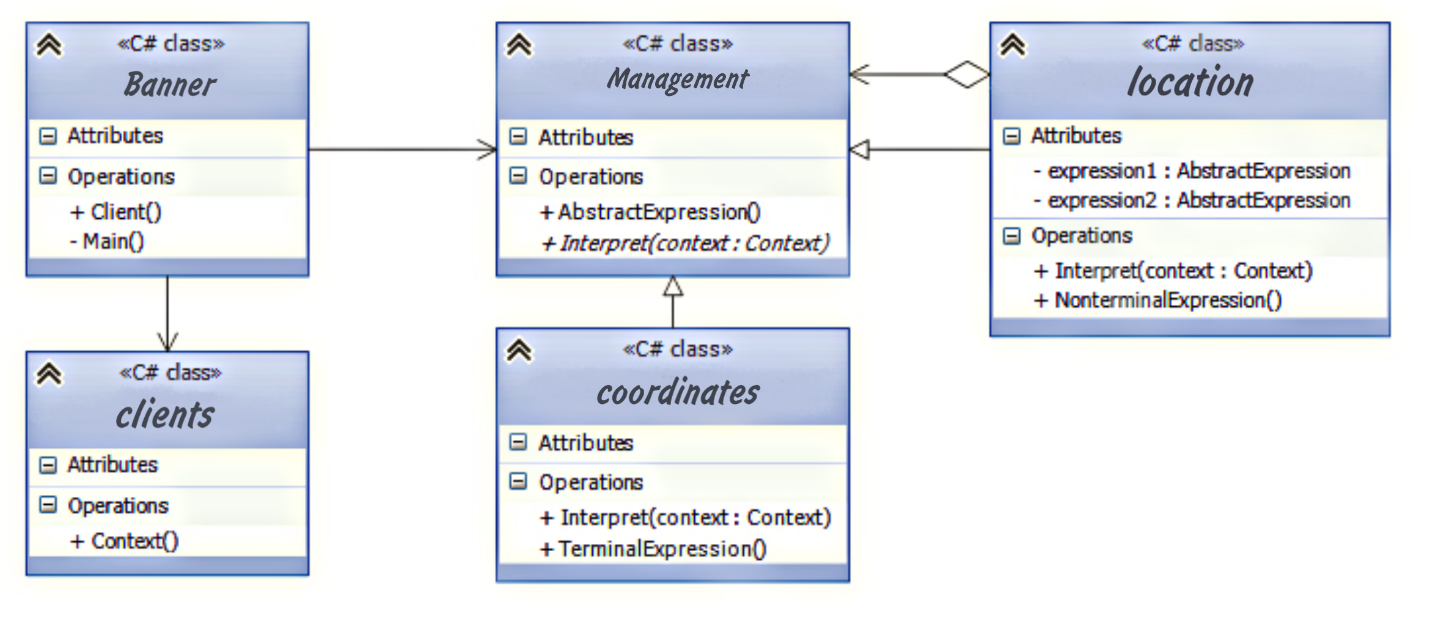


**4. UML Diagrams**

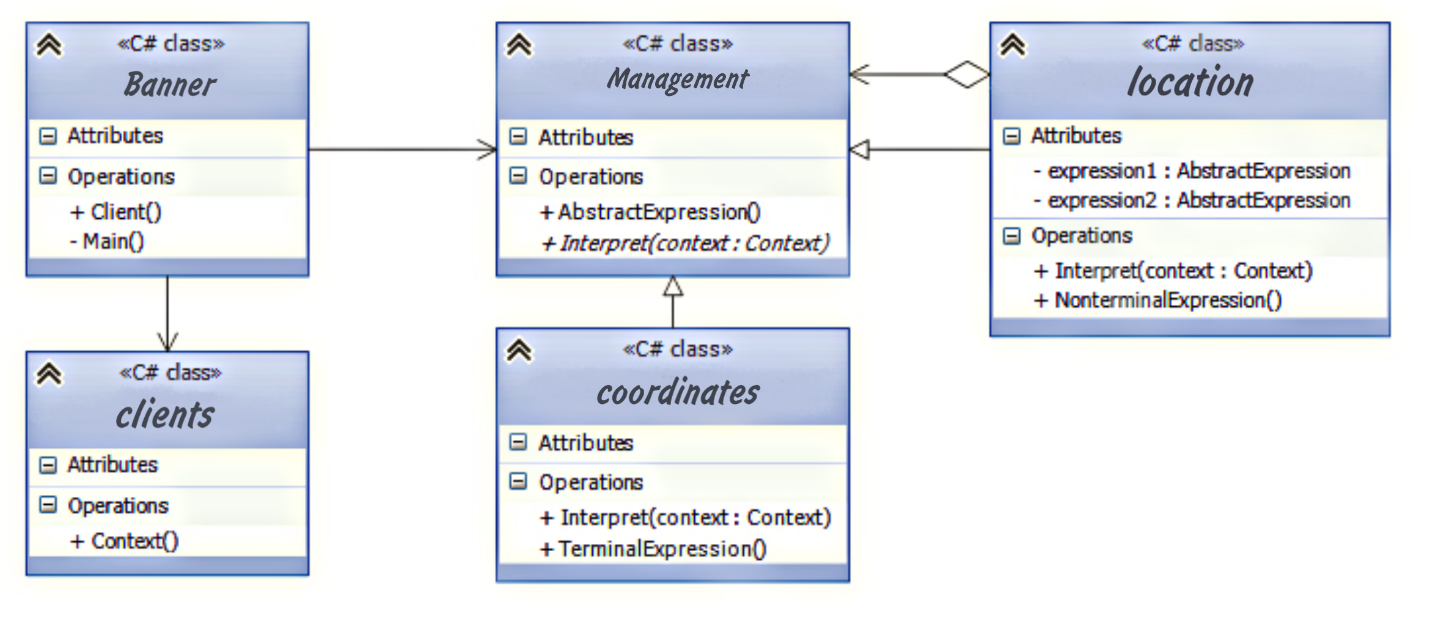
Case 8.



Case 8



Case 9

****

**Conclusion**

During developing software, programmers need to think carefully about the architecture, breaking it down into separate, simpler blocks and subtasks. Developer can think “What if someone has already solved this problem?” and “How to make it easier?”. The fact is that almost certainly similar problems have already been solved before, and there are already well-thought-out solutions drawn up by experts. Often, when developing software, design patterns or patterns are used, which represent a certain model of interaction of classes to solve a problem.

Implementing a project using patterns greatly simplifies the development process. The main advantage of templates is the availability of ready-made abstract solutions for most problems. Moreover, each type of task has its own template with a name, which simplifies communication between developers, increases the readability of the code and understanding of the project architecture.

Since the modules and elements of the project are unified using templates, the number of errors is significantly reduced.

It is important to keep in mind, that a blind desire to use design patterns for no reasons can, on the contrary, complicate the design of the project and negatively affect performance.

There are several types of design patterns, each of which is designed to solve a specific type of problem:

- Generative patterns. Create new objects in the system. (Builder from my case)

- Structural patterns. Solve layout problems based on classes and objects.

- Behavioral patterns. Distribute responsibilities between system objects. (Strategic, Integrator from my cases)

I used three types of patterns for three cases from the types of Behavioral and Generative patterns.